Recursos para optimizar los parsers de NLTK

Fernando Carranza
fernandocarranza86@gmail.com
Catalina Rubio
catarubio89@gmail.com
Fernando Schiaffino
schiaffinofernando@gmail.com

Coloquio "Lingüística computacional y teoría gramatical"
XVIII Congreso de la Sociedad Argentina de Estudios Lingüísticos
Universidad Nacional del Comahue, Gral. Roca
10 a 13 de mayo de 2023

Presentación

En esta presentación vamos a presentar algunos recursos que diseñamos para enriquecer los parsers de gramáticas basadas en rasgos que posee NLTK. Estos recursos se idearon para solucionar dos problemas que puede tener NLTK a la hora de utilizar sus parsers para procesar lenguaje natural:

1. El excesivo costo de procesamiento
2. La falta de robustez

Estructura

El trabajo tiene la siguiente estructura.

1. Introducción
2. NLTK
   - Presentación general de la librería
   - Los parsers de NLTK
3. Recursos propuestos
   - Estrategia para darle robustez a la gramática
   - Estrategia para disminuir el costo de procesamiento
4. Palabras finales

Caracterización general de NLTK

- NLTK (ver Bird 2006 y Bird *et al.* 2009, `https://www.nltk.org/`) es una librería de procesamiento del lenguaje natural.
- Está diseñada principalmente para ser usada como recurso pedagógico. Fue programada en Python por su facilidad de aprendizaje.
- Es una librería que busca la facilidad de lectura y la consistencia, antes que la rapidez a la hora de correr el algoritmo.

Ventajas y desventajas de NLTK

- Dada su fácil legibilidad, el hecho de que sea de código libre y la variedad de módulos para distintas tareas que posee, NLTK se ha vuelto una librería muy popular. Actualmente el código cuenta con contribuciones de 380 personas y forma parte de las dependencias de 189.447 repositorios en Github.
- Sin embargo, el hecho de que NLTK esté orientada principalmente a la enseñanza hace que esta librería, a diferencia de otras que están orientadas a tareas generales de procesamiento del lenguaje natural, como Stanza (Qi *et al.* 2020, `https://stanfordnlp.github.io/stanza/`) o Spacy (`https://spacy.io/`), no sea lo suficientemente óptima como para su utilización en escenarios menos controlados.

Organización de la librería

La librería está organizada en torno a distintos paquetes, cada uno de los cuales se especializa, a grandes rasgos, en alguna tarea de procesamiento del lenguaje natural. Estos paquetes contienen a su vez módulos, que implementan diferentes algoritmos para resolver la tarea en cuestión. En `https://www.nltk.org/py-modindex.html` aparece el índice completo de paquetes y módulos. Entre ellos, se incluyen módulos para las siguientes tareas:

- tokenización
- stemming
- postagging
- análisis de sentimiento
- parsing

En lo que respecta a este trabajo, nos interesan particularmente los módulos que se ocupan del parsing.

Criterios de clasificación de parsers

- El parsing consiste en un algoritmo que toma una cadena x y una gramática G e idealmente devuelve como resultado al menos un análisis estructural para x si $x \in L(G)$.
- Los parsers se pueden clasificar de acuerdo a al menos dos criterios:
  - su funcionamiento interno
  - el tipo de gramáticas con las que son compatibles

Funcionamiento interno

A lo largo de la historia del Procesamiento del Lenguaje natural se han propuesto distintos tipos de algoritmos generales de parsing. Entre ellos, los siguientes (ver Roark y Sproat 2007):

- **Algoritmos deterministicos**
  - Shift reduce parser
  - Push-down parsers (e.g., recursive descent parser)
  - Left corner parser
- **Algoritmos no deterministicos**
  - Extensiones de los deterministicos
    - Re-analysis
    - Beam-search
  - Programación dinámica (chart parsers)
    - CYK Parser (Kasami 1965, Younger 1967, Cocke y Schwartz 1970, Kaplan 1973)
    - Algoritmo de Earley (Earley 1970)
    - Viterbi

Tipos de gramáticas

A su vez, los parsers se pueden clasificar respecto del tipo de gramáticas para las que están diseñados. Existen tres grandes familias de gramáticas en lo que respecta al modo en que conciben la estructura de las oraciones:

- **Gramáticas basadas en la noción de constituyente**: La estructura de las oraciones consiste en agrupamientos de palabras jerárquicamente organizados.
- **Gramáticas basadas en la noción de dependencia**: La estructura de las oraciones consiste en palabras que establecen relaciones de dependencia entre núcleos y dependientes (ver Carranza 2016).
- **Gramáticas categoriales**: La estructura de las oraciones consiste en sucesivas deducciones que se realizan a partir de un conjunto de reglas de deducción aplicadas a premisas que tienen la forma de funciones o argumentos (ver McGee Wood 1993).

Parsers de NLTK según funcionamiento interno

Con respecto al primer criterio de clasificación, NLTK provee, en línea con sus objetivos didácticos, diversos módulos que implementan los distintos tipos de parsers, algunos de los cuales proveen incluso interfaces gráficas para entender mejor cómo funciona el algoritmo. Algunos de estos son los siguientes:

- Recursive Descent (posee interfaz gráfica)
- Shift reduce (posee interfaz gráfica)
- Chart Parser (posee interfaz gráfica)
- Viterbi parser

Parsers de NLTK según tipo de gramática

Con respecto al segundo criterio de clasificación, NLTK provee herramientas para los tres tipos de gramáticas:

- Las gramáticas categoriales están implementadas en un paquete especial, nltk.ccg (`https://www.nltk.org/api/nltk.ccg.html#module-nltk.ccg`) que posee su propio módulo de parseo.
- Dentro del paquete nltk.parse, hay dos módulos especialmente dedicados a gramáticas basadas en dependencias tanto proyectivas como no proyectivas.
- La mayor parte de los parsers de ntlk.parse están hechos para ser usados con gramáticas basadas en constituyentes.

Puesto que las gramáticas basadas en constituyentes han sido las privilegiadas en la tradición lingüística, nos limitaremos a este tipo de gramáticas.

- Las gramáticas basadas en constituyentes constituyen una familia, de entre las cuales las más conocidas son las gramáticas independientes de contexto. Desde Shieber (1985), se sabe que las lenguas naturales no pueden formalizarse enteramente mediante gramáticas independientes de contexto.
- Para extender el poder expresivo, se suele usar gramáticas enriquecidas con rasgos. Por este motivo, nos concentramos en este tipo de gramáticas.
- NLTK posee soporte para parsear oraciones con gramáticas basadas en constituyentes enriquecidas con rasgos, que se guardan en archivos de extensión fcfg.

Ejemplo de gramática enriquecida con rasgos:

```
S -> SN[PER=?x, NUM=?n] SV[PER=?x, NUM=?n]
SN[PER=1, NUM=sg] -> 'yo'
SN[PER=2, NUM=sg] -> 'vos'
SN[PER=2, NUM=pl] -> 'ustedes'
SV[PER=?x, NUM=?n] -> V[PER=?x, NUM=?n]
V[PER=1, NUM=sg] -> 'hablo'
V[PER=2, NUM=sg] -> 'hablás'
V[PER=2, NUM=pl] -> 'hablan'
```

Ahora bien, el parseo de NLTK para gramáticas basadas en rasgos enfrenta dos grandes problemas:

- No provee gramáticas robustas que le permitan usarlo en escenarios más realistas.
- En el caso de proveerle una gramática lo suficientemente robusta, el costo de procesamiento se dispara y el parser se tilda.

Los recursos que presentamos en este trabajo pretenden sortear estos dos problemas. Estos recursos pueden encontrarse en `https://github.com/barracudanlp/QuarantineGrammars`

Las reglas de las gramáticas basadas en rasgos se pueden dividir en dos grandes grupos:

- Las reglas de estructura de frase:

  `S -> SN SV`

- Las reglas de inserción léxica:

  `N -> mesa`

Naturalmente, para darle robustez a la gramática es preciso enriquecer ambos grupos. Puesto que el segundo grupo es el más numeroso, en este trabajo privilegiamos ese. Afortunadamente, existen recursos de acceso libre que sirven para eso. Por ejemplo, Freeling (Padró y Stanilovsky 2012, Padró *et al.* 2010, `https://nlp.lsi.upc.edu/freeling/node/1`), una librería de procesamiento del lenguaje natural, posee en su repositorio de git (`https://github.com/TALP-UPC/freeling`) una serie de diccionarios de español que parea cada forma de palabra con su respectivo lema y etiqueta de clase de palabra (*pos-tag*). Las etiquetas responden al tagset de EAGLES (Leech y Wilson 1996).

Algunos ejemplos de formas de palabra, lema y anotación morfosintáctica con el tagset de EAGLES para algunos nombres:

```
perro perro NCMS000
perros perro NCMP000
persa persa NCCS000
persas persa NCCP000
persecuciones persecución NCFP000
persecución persecución NCFS000
```

NOMBRES			
Pos.	**Atributo**	**Valor**	**Código**
1	Categoría	Nombre	N
2	Tipo	Común	C
		Propio	P
3	Género	Masculino	M
		Femenino	F
		Común	C
4	Número	Singular	S
		Plural	P
		Invariable	N
5-6	Clasificación semántica	Persona	SP
		Lugar	G0
		Organización	O0
		Otros	V0
7	Grado	Aumentativo	A
		Diminutivo	D

Entre nuestros recursos incluimos un script de bash que toma estas entradas léxicas para usarlas como insumo para la construcción de las reglas de inserción léxica. Para transformarlas al formato que precisa la gramática basada en rasgos, elaboramos un json para mapear las etiquetas de eagles con los valores de las categorías y los rasgos que debería tener cada entrada léxica.

En las siguientes diapositivas se ejemplifican las correspondencias de EAGLES con nuestros rasgos para el caso de los verbos y los nombres:

```
"V":{   "subcategoria":{"posicion": 1},
     "modo": {"posicion": 2,
          "I": "IND",
          "S": "SUBJ",
             "M": "IMP",
         "N": "INF",
         "G": "GDIO",
          "P":"PPIO"},
     "tiempo": { "posicion": 3,
         "P":"presente",
         "I": "imperfecto",
            "F": "futuro",
          "S": "pasado",
           "C": "condicion",
```

```
                "0": "atemp"},
        "persona":{ "posicion": 4},
         "numero":{   "posicion": 5,
              "S":"sg",
              "P": "pl"}},
"N":{"subcategoria":{   "posicion": 1,
             "C": "COMUN",
                "P": "PROPIO"},
        "genero": { "posicion": 2,
            "M": "masc",
             "F": "fem",
             "C": "?gen"},
        "numero": { "posicion": 3,
             "S":"sg",
             "P": "pl",
            "N": "inv"},
        "sema": {    "posicion": 4,
```

```
        "S": "persona",
       "G": "lugar",
        "V": "otros"},
"semb": { "posicion": 5,
        "P": "persona",
        "O": "_"},
"grado": { "posicion": 6,
    "A":"aum",
    "D": "dim"}}
```

Por ejemplo, a partir de la siguiente información que provee el diccionario de Freeling para la palabra perro:

```
perro perro NCMS000
```

Construye la siguiente entrada de inserción léxica:

```
N[GEN='masc', NUM='sg', SEM=<\x.perro(x)>] perro
```

- Hicimos la prueba de darle como input al chart parser de NLTK una gramática de más de mil reglas (principalmente, reglas de inserción léxica y unas pocas de estructura de frase) y oraciones de una y dos palabras que el parser debería reconocer. El resultado fue que el parser se tildó por el excesivo costo de procesamiento, aun cuando la longitud de la oración era tan reducida.
- Para solucionar este problema, hicimos un script en Python para que la gramática que toma el parser se elaborara directamente a partir de los tokens que aparecen en la oración. La gramática en cuestión se construye a partir de las reglas de estructura de frase, que son estables, y de las reglas de inserción léxica que involucren tokens que efectivamente aparecen en la oración. Mediante este recurso, el parser corre en un tiempo asequible computacionalmente.

Salida del parser frente a la oración “el perro mira”.

```
(S[SEM=<(exists e.agente(e,iota x.(sg(x) & perro(x)))
 & mirar(e) & presente(e))>]
    (NP[GEN='masc', NUM='sg', PER=?m,
    SEM=<\Q.Q(iota x.(sg(x) & perro(x)))>]
        (D[GEN='masc', NUM='sg',
        SEM=<\P Q.Q(iota x.(sg(x) & P(x)))>] el)
        (N[GEN='masc', NUM='sg', SEM=<\x.perro(x)>] perro))
     (VP[MODE='IND', NUM='sg', PER=3, +PRINC,
     SEM=<\e.(mirar(e) & presente(e))>,
     TENSE='presente']
        (Verb[MODE='IND', NUM='sg', PER=3,
        SEM=<\e.(mirar(e) & presente(e))>,
        SUBCAT='M', TENSE='presente']
            (V[LEMA='mirar', MODE='IND', NUM='sg',
            PER=3, SEM=<\e.(mirar(e) & presente(e))>,
           SUBCAT='M', TENSE='presente'] mira))))
```

En este trabajo hemos presentado algunos recursos que elaboramos para enriquecer el parseo de gramáticas basadas en rasgos de NLTK, atendiendo específicamente a dos problemas: el de la falta de gramáticas lo suficientemente robustas y el del excesivo costo de procesamiento que tiene NLTK.

Bibliografía I

Bird, S. (2006). Nltk: the natural language toolkit. En *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 69–72.

Bird, S., Klein, E., y Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media.

Carranza, F. (2016). Tesnière y su gramática de dependencias: continuidades y discontinuidades. *RAHL: Revista argentina de historiografía lingüística*, 8(2):59–78.

Cocke, J. y Schwartz, J. T. (1970). Programming languages and their compilers: Preliminary notes. Technical report, New York.

Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Kaplan, R. M. (1973). A general syntactic processor. En Rustin, R., editor, *Natural Language Processing*, pp. 193–241. Algorithmics Press.

Bibliografía II

Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. technical report afcrl-65-758. Technical report, Bedford.

Leech, G. y Wilson, A. (1996). *Recommendations for the Morphosyntactic Annotation of Corpora*.

McGee Wood, M. (1993). *Categorial Grammars*. Routledge, London/New York.

Padró, L., Collado, M., Reese, S., Lloberes, M., y Castellón, I. (2010). Freeling 2.1: Five years of open-source language processing tools. En *Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010)*, La Valletta. ELRA.

Padró, L. y Stanilovsky, E. (2012). Freeling 3.0: Towards wider multilinguality. En *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul. ELRA.

Bibliografía III

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., y Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. En *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Roark, B. y Sproat, R. (2007). *Computational Approaches to Morphology and Syntax*. Oxford University Press, Oxford.

Shieber, S. (1985). Evidence against the context-freeness of natural language. (8):333–343.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.